**WHAT WE CLAIM IS:**

1. A processor, for executing instructions of a program stored in compressed form in a program memory, comprising:

a program counter which identifies a position in said program memory;

an instruction cache, having a plurality of cache blocks, each for storing one or more instructions of said program in decompressed form;

a cache loading unit, comprising a decompression section, operable to perform a cache loading operation in which one or more compressed-form instructions are read from said position in the program memory identified by the program counter and are decompressed and stored in one of said cache blocks of the instruction cache;

a cache pointer which identifies a position in said instruction cache of an instruction to be fetched for execution;

an instruction fetching unit which fetches an instruction to be executed from the position identified by the cache pointer and which, when a cache miss occurs because the instruction to be fetched is not present in the instruction cache, causes the cache loading unit to perform said cache loading operation; and

an updating unit which updates the program counter and cache pointer in response to the fetching of instructions so as to ensure that said position identified by said program counter is maintained consistently at the position in said program memory at which the instruction to be fetched from the instruction cache is stored in compressed form.

2. A processor as claimed in claim 1, wherein said position in the instruction cache of an instruction to be fetched is identified by said cache

pointer in terms of an imaginary address assigned to the instruction, at which the instruction is considered to exist when held in decompressed form in one of said cache blocks.

3. A processor as claimed in claim 2, wherein said imaginary address of an instruction is assigned thereto during assembly/linking of said program based on the sequence of original instructions in the program prior to compression.

4. A processor as claimed in claim 2, wherein imaginary address information, from which said imaginary address assigned to each instruction is derivable, is stored with the compressed-form instructions in the program memory and is employed in the cache loading operation so as to associate with each decompressed instruction present in the instruction cache the imaginary address assigned thereto.

5. A processor as claimed in claim 1, wherein:

the compressed-form instructions are stored in the program memory in one or more compressed sections, the compressed-form instructions belonging to each section occupying one of said cache blocks when decompressed, and at least one section also contains imaginary address information relating to the instructions belonging to the section; and

said cache loading unit is operable, in said cache loading operation, to decompress and load into one of said cache blocks one such compressed section stored at the position in the program memory identified by the program counter.

6. A processor as claimed in claim 5, wherein said imaginary address information of said one section specifies the imaginary address at which a first one of the decompressed instructions corresponding to the compressed section is considered to exist when the

decompressed instructions are held in one of the cache blocks.

7. A processor as claimed in claim 5, wherein in said cache loading operation the cache block into which the decompressed instructions of the compressed section are loaded is assigned an imaginary block address based on said imaginary address assigned to an instruction contained in the section.

8. A processor as claimed in claim 7, wherein each said cache block has an associated cache tag in which is stored said imaginary block address assigned to the cache block with which the cache tag is associated.

9. A processor as claimed in claim 5, wherein said imaginary address information is contained in only a first one of said compressed sections to be loaded.

10. A processor as claimed in claim 5, wherein each said compressed section contains imaginary address information relating to the instructions belonging to the section concerned.

11. A processor as claimed in claim 5, wherein the or each said compressed section further contains a decompression key which is employed by said decompression section to effect the decompression of the instructions belonging to the compressed section during the cache loading operation.

12. A processor as claimed in claim 11, wherein the instructions of said program include, prior to compression, preselected instructions that are not stored explicitly in any said compressed section, and the decompression key of the or each compressed section identifies the positions at which the preselected instructions are to appear in the cache block when the compressed section is decompressed.

13. A processor as claimed in claim 12, wherein said preselected instructions are "no operation"

instructions.

14. A processor as claimed in claim 5, wherein said updating unit comprises:

a next-section locating section operable, in the event of such a cache miss, to employ next-section-locating information, stored in association with the cache block which was accessed most recently to fetch an instruction, to locate the position in the program memory of a next compressed section following the compressed section corresponding to that most-recently-accessed cache block.

15. A processor as claimed in claim 14, wherein such next-section-locating information is stored in association with each cache block in which valid decompressed instructions are held, the stored information being for use in locating the position in the program memory of the next compressed section following the compressed section corresponding to the cache block concerned.

16. A processor as claimed in claim 15, wherein the next-section-locating information is stored in association with each cache block when that block is loaded in such a cache loading operation.

17. A processor as claimed in claim 14, wherein said next-section-locating information associated with the cache block relates to a size of the compressed section corresponding to that cache block.

18. A processor as claimed in claim 17, wherein said size is determined by the cache loading unit when loading the cache block in the cache loading operation.

19. A processor as claimed in claim 15, wherein the updating unit comprises:

a locating information register section which stores said next-section-locating information associated with the most-recently-accessed cache block; and

a copying section operable, when an instruction held in one of the cache blocks is fetched, to copy into the locating information register section said next-section-locating information stored in association with that block;

said next-section-locating section being operable, in the event of such a cache miss, to employ the next-section-locating information stored in the location information register section to locate said position of said next compressed section.

20. A processor as claimed in claim 12, wherein: said updating unit comprises:

a next-section locating section operable, in the event of such a cache miss, to employ next-section-locating information, stored in association with the cache block which was accessed most recently to fetch an instruction, to locate the position in the program memory of a next compressed section following the compressed section corresponding to that most-recently-accessed cache block; and

said next-section-locating information associated with the cache block represents the number of instructions held in that cache block that are not said preselected instructions.

21. A processor as claimed in claim 20, wherein the decompression section comprises a counter operable, during such a cache loading operation, to count the number of decompressed instructions that are not said preselected instructions.

22. A processor as claimed in claim 1, operable to execute a hardware-controlled loop, wherein:

said updating unit further comprises respective first and second loop control registers and operates, upon initiation of execution of such a hardware-controlled loop, to cause the program-counter value to be stored in said first loop control register and to

cause the cache-pointer value to be stored in said second loop control register, and further operates, upon commencement of each iteration of the loop after said first iteration thereof, to reload said program counter with the value held in said first loop control register and to reload said cache pointer with the value held in said second loop control register.

23. A processor as claimed in claim 1, wherein the instructions of said program comprise very-long-instruction-word (VLIW) instructions.

24. A processor as claimed in claim 1, wherein the updating unit is operable, when an interrupt occurs during execution of a program, to cause the program-counter value and cache-pointer value to be saved pending handling of the interrupt, and, when the execution of the program is resumed, to cause the saved values to be restored in the program counter and cache pointer.

25. A processor as claimed in claim 14, wherein the updating unit is operable, when an interrupt occurs during execution of a program, to cause said next-section locating information associated with the most-recently-accessed cache block to be saved pending handling of the interrupt, and, when execution of the program is resumed, to cause the saved next-section locating information to be restored.

26. A processor as claimed in claim 20, wherein the updating unit is operable, when an interrupt occurs during execution of a program, to cause the values held in said loop control registers to be saved pending handling of the interrupt, and, when execution of the program is resumed, to cause the saved values to be restored in the loop control registers.

27. A method of compressing a program to be executed by a processor in which compressed-form instructions stored in a program memory are

decompressed and cached in an instruction cache prior to being issued, the method comprising:

converting a sequence of original instructions of the program into a corresponding sequence of such compressed-form instructions;

assigning such original instructions imaginary addresses according to said sequence thereof, the assigned imaginary addresses being imaginary addresses at which the instructions are to be considered to exist when held in decompressed form in said instruction cache of the processor; and

storing, in said program memory, the compressed-form instructions together with imaginary address information specifying said assigned imaginary addresses so that, when the compressed-form instructions are decompressed and loaded by the processor into the instruction cache, the processor can assign the specified imaginary addresses to the decompressed instructions.

28. A method as claimed in claim 27, wherein the assigned imaginary addresses are selected so that instructions likely to coexist in the instruction cache at execution time will not be mapped to the same cache block.

29. A method as claimed in claim 27, wherein the compressed-form instructions are stored in said program memory in one or more compressed sections, the compressed-form instructions belonging to each section occupying one cache block of the processor's instruction cache when decompressed, and at least one compressed section also containing imaginary address information relating to the instructions of that section.

30. A method as claimed in claim 29, wherein said imaginary address information specifies the imaginary address at which a first one of the decompressed

instructions corresponding to said one compressed section is to be considered to exist when the decompressed instructions are held in said instruction cache.

31. A method as claimed in claim 29, wherein said imaginary address information is contained in only a first one of said compressed sections to be loaded.

32. A method as claimed in claim 29, wherein each said compressed section contains imaginary address information relating to the instructions belonging to the section concerned.

33. A method as claimed in claim 29, wherein the or each said compressed section further contains a decompression key for use by the processor to carry out the decompression of the instructions belonging to said section.

34. A method as claimed in claim 33, wherein said sequence of original instructions of the program comprises preselected instructions that are not stored explicitly in any said compressed section, and the decompression key of the or each said compressed section identifies the positions at which said preselected instructions exist are to appear in a decompressed sequence of instructions corresponding to the section.

35. A method as claimed in claim 34, wherein said preselected instructions are "no operation" instructions.

36. A computer-readable recording medium storing a computer program which carries out a method of compressing a processor program to be executed by a processor, the processor being operable to decompress compressed-form instructions stored in a program memory and to cache the decompressed instructions in an instruction cache prior to issuing them, the computer program comprising:

a converting portion which converts a sequence of original instructions of the processor program into a corresponding sequence of such compressed-form instructions;

an assigning portion which assigns such original instructions imaginary addresses according to said sequence thereof, the assigned imaginary addresses being imaginary address at which the instructions are to be considered to exist when held in decompressed form in said instruction cache of the processor; and

a storing portion which stores, in said program memory, the compressed-form instructions together with imaginary address information specifying said assigned imaginary addresses so that, when the compressed-form instructions are decompressed and loaded by the processor into the instruction cache, the processor can assign the specified imaginary addresses to the decompressed instructions.

37. A processor, for executing instructions of a program stored in compressed form in a program memory, comprising:

a program counter for identifying a position in said program memory;

an instruction cache, having a plurality of cache blocks, each for storing one or more instructions of said program in decompressed form;

cache loading means, including decompression means, operable to perform a cache loading operation in which one or more compressed-form instructions are read from said position in the program memory identified by the program counter and are decompressed and stored in one of said cache blocks of the instruction cache;

a cache pointer for identifying a position in said instruction cache of an instruction to be fetched for execution;

instruction fetching means for fetching an

instruction to be executed from the position identified by the cache pointer and operable, when a cache miss occurs because the instruction to be fetched is not present in the instruction cache, to cause the cache loading means to perform such a cache loading operation; and

updating means for updating the program counter and cache pointer in response to the fetching of instructions so as to ensure that said position identified by said program counter is maintained consistently at the position in said program memory at which the instruction to be fetched from the instruction cache is stored in compressed form.